TECHNOLOGY

QUÈ HI HA DE NOU?

“DOCKER COM A PROCÉS, PAAS COM A MÀQUINA, ARQUITECTURA DE MICROSERVEIS COM A MODEL DE PROGRAMACIÓ.”

L’estil d’arquitectura en microserveis remarca l’augment d’abstraccions en el món dels desenvolupadors a causa de la proliferació de contenidors i de l’èmfasi en l’acoblament baix, el qual ofereix un gran nivell d’aïllament operacional. Els desenvolupadors poden veure els contenidors com a processos independents i el PaaS (plataforma com a servei) com l’objectiu de desplegament comú i utilitzant l’arquitectura de microserveis com a estil comú. Desacoblar l’arquitectura permet el mateix pels equips, facilitant la coordinació entre ells. El seu atractiu tant per a desenvolupadors com per a DevOps ha fet que s’hagi convertit en el nou estàndard de desenvolupament a moltes organitzacions.

SOBRE EL RADAR TECNOLÒGIC

Als treballadors de ThoughtWorks els apassiona la tecnologia. La creem, l'investiguem, la posem a prova, la compartim gràcies al codi obert, escrivim sobre ella i sempre la volem millorar per a beneficiar a tothom. La nostra missió és arribar a l'excel•lència i revolucionar la informàtica. Amb aquesta missió en ment, creem i compartim el Radar Tecnològic. El Radar el crea el gabinet d'assessors de ThoughtWorks Technology, un grup de líders sèniors de ThoughtWorks. Es reuneixen periòdicament per a parlar sobre l'estratègia global de ThoughtWorks i les tendències tecnològiques que tenen un impacte significatiu sobre la nostra industria.

Aquest Radar captura l'essència de les xerrades del gabinet d'assessor i la comparteix en un format adequat per a una gran varietat d'actors, des de gerents de sistemes (CIO) fins a desenvolupadors. Com que entenem el Radar com un resum, recomanem que s'explorin més detalladament aquestes tecnologies. El Radar té una naturalesa més aviat gràfica i agrupa els elements en tècniques, eines, plataformes i llenguatges i entorns de treball. Si algun element pot aparèixer en més d'un quadrant, escollim el que ens sembla més adient. Agrupem aquests elements en 4 anells per a reflectir el que en pensem. Els anells són:

Creiem fermament que la industria hauria d'adoptar aquests elements. Nosaltres els

Els elements nous o que han canviat de manera significativa des de l'últim Radar es representen amb un triangle mentre que els elements que no s'han mogut es representen amb un cercle. Ens interessen molts més temes dels que podem posar en un document d'aquesta mida per la qual cosa hem deixat esvair molts elements del Radar anterior per a fer lloc als nous elements. Que ja no hi sigui no vol dir que ja no ens interessi.

Per a més informació sobre el Radar, visiteu thoughtworks.com/radar/faq

Per aquesta edició hem decidit recuperar les **proves de contracte guiats pel consumidor** tot i que anteriorment vam deixar que s'esvaís. El concepte no és nou però, ara que els microserveis estan sent acceptats per la majoria, hem de recordar que els contractes guiats pels consumidors són una part essencial d'un portafoli madur de proves de microserveis que permeti els desplegaments independents de serveis. Volem remarcar, a més, que les proves de contractes guiats pels consumidors són una tècnica i una actitud que no requereixen que s'implementi cap eina especial. Ens encanten els entorns de treball com el Pact perquè permeten que es puguin implementar amb més facilitat les proves de contracte en certs contextos. Hem vist, però, una tendència per la qual els equips es centren més en l'entorn de treball que en la pràctica general. Dur a terme proves Pact no garanteix que s'estiguin creant contractes guiats pels consumidors. A més, en moltes situacions s'haurien de crear bons contractes guiats pels consumidors fins i tot quan no hi ha cap eina de proves implementada.

Els equips estan buscant la automatització en els seus entorns i, fins i tot, en la seva estructura de desenvolupament. **Pipelines as code** defineix el desplegament de canals via codi en lloc de configurar una eina CI/CD en funcionament. LambdaCD, Drone, GoCD i Concourse són exemples d'eines que permeten l'ús d'aquesta tècnica. A més, eines d'automatització de la configuració per a sistemes CI/CD, com GoMatic, es poden utilitzar per tractar el canal de desplegament com a codi (adaptat i provat).

Les empreses han adoptat massivament les API com a la manera de demostrar les capacitats de l'empresa tant per a desenvolupadors interns com externs. Les API permeten experimentar sense esforços amb noves idees de l'empresa recombinant els elements essencials. Aleshores, però, què diferencia una API d'un servei ordinari d'integració d'empresa (EIS)? Una de les diferències és que es tracta les **API com un producte**, fins i tot quan el consumidor és un sistema intern. Els equips que creen APIs haurien d'entendre les necessitats dels seus clients i fer que el producte sigui adequat per a ells. Els productes, a més, es milloren, es mantenen i es dóna suport durant molt de temps. Haurien de tenir un propietari que defensi el consumidor i té com objectiu la millora constant. Els productes es mantenen i suporten activament i són fàcils

les seves pàgines i característiques i cada una d'aquestes divisions és responsabilitat d'un sol equip. Hi ha diverses tècniques (algunes noves i d'altres velles) per aconseguir que les característiques de l'aplicació es complementin per crear una millor experiència d'usuari però l'objectiu segueix sent el de permetre que es pugui desenvolupar, provar i desplegar cada característica independentment de les altres. L'enfocament de motor d'interfície (BFF per les seves sigles en anglès) funciona molt bé aquí on cada equip desenvolupa un BFF per a donar suport a la seva col•lecció de característiques de l'aplicació.

Amb l'augment de la popularitat del patró de motor d'interfície (BFF) i l'ús d'entorns de treball d'associació de dades com React.js, hem vist una resposta negativa cap a les arquitectures REST. Els crítics acusen REST de crear interaccions inefcients entre sistemes i de no adaptar-se a l'evolució de les necessitats dels clients. Proposen com a mecanismes d'obtenció de dades alternatius entorns de treball com GraphQL o Falcor, ja que permeten que el client especifqui el format de les dades retornades. En la nostra experiència, però, no és REST el que causa aquests problemes sinó que més aviat apareixen perquè no s'ha sabut modelar el domini com una col•lecció de recursos. Desenvolupar nativament serveis que només exposen models de dades estàtics i jeràrquics a través de URLs basades en plantilles té com a resultat una **implementació anèmica de REST**. En un domini amb un model ric, el REST hauria de permetre més que una simple recol•lecció de dades repetitiva. En una arquitectura RESTful totalment evolucionada, els esdeveniments d'empresa i els conceptes abstractes es modelen també com a recursos i la implementació hauria d'utilitzar correctament l'hipertext, la relació d'enllaços i els tipus de contingut multimèdia per a maximitzar l'acoblament entre els serveis. Aquest anti-patró està estretament relacionat amb el patró de Model de Domini Anèmic i té com a resultat serveis que ocupen llocs baixos en el Model de Maduresa de Richardson. Disposem de més recomanacions sobre el disseny de APIs de RESt efectives en el nostre article sobre troballes.

Seguim veient com organitzacions busquen tecnologies "cool" les quals els fan prendre riscos i complicar-se la

Actualment, la seguretat de transport HTTP estricta (**HSTS**) és una política vastament suportada que permet que les pàgines web es protegeixin contra atacs degradants. Un atac degradant, en el context de HTTPS, és el que comporta que els usuaris del servei acabin a HTTP en lloc de HTTPS, el que permet nous atacs com ara els atacs man-in-the-middle. Amb l'HSTS, el servidor informa al navegador, per mitjà d'una capçalera, que només hauria d'utilitzar HTTPS per accedir a la pàgina web. El suport per part dels navegadors ja és prou generalitzat com per què aquesta funció de fàcil implementació s'afegeixi a qualsevol pàgina web que utilitzi HTTPS. L'Observatori de Mozilla pot ajudar a identificar aquesta i d'altres capçaleres útils i opcions de configuració que millorin la privacitat i la seguretat. Quan s'implementi HSTS, és de vital importància verificar que

PLATAFORMES *continuació*

d'integració, la varietat de protocols i connectors que suporta i la gran API de gestió.

Els nostres equips segueixen gaudint de **AWS Lambda** i l'estan començant a utilitzar per a experimentar amb arquitectures sense servidor, combinant Lambda amb API Gateway

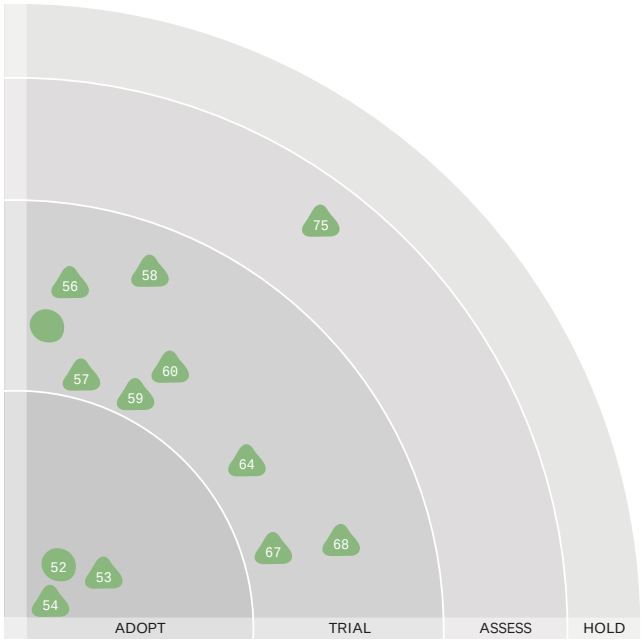
PLATAFORMES *continuació*

EINES

Babel.js s'ha convertit en el compilador per defecte per escriure JavaScript d'última generació. El seu ecosistema s'està desenvolupament gràcies al seu sistema de connectors reestructurat. Permet que els desenvolupadors escriguin codi ES6 (i fins i tot ES7) que corre al navegador o al servidor sense sacrificar la retrocompatibilitat amb navegadors antics i pel qual cal molt poca configuració. Té un suport de primer nivell per a sistemes de desenvolupament i prova per la qual cosa es pot incorporar sense problemes a processos de treball ja existents. És un gran programari que ha facilitat l'adopció i la innovació de ES6 (i ES7).

Quan es combinen tècniques i estils d'arquitectura moderns, com els microservices, DevOps i el control de qualitat durant el desenvolupament, els equips de desenvolupament necessiten eines de supervisió cada vegada més sofisticades. Ja no és suficient mirar un gràfic sobre l'ús de la CPU i del disc. És per això que molts equips utilitzen eines com Graphite o Kibana per a recol•lectar mètriques sobre aplicacions i empreses. **Grafana** facilita la creació de panells d'instruments útils i elegants per a dades obtingudes de diverses fonts. Existeix una funció especialment útil que permet sincronitzar els temps de diferents gràfics per identificar correlacions en les dades. El sistema de plantilles que s'ha afegit té molt potencial i farà que la gestió de serveis similars sigui encara més fàcil. Grafana s'ha convertit en la nostra elecció en aquesta categoria gràcies a les seves funcions.

Les imatges de màquina s'estan convertint en un canal de desplegament bàsic i existeixen diverses eines i tècniques per crear-les. Nosaltres recomanem **Packer** abans que d'altres alternatives per la gran quantitat de funcions i les experiències positives que hem tingut. També recomanem no escriure scripts propis per fer el que Packer pot fer per defecte.



d'eines i múltiples canals el fan un ingredient clau per a l'entrega continua per a mòbils.

Provar que la disposició i l'estil de pàgines web funciona correctament en diferents mides pot ser un procés lent i, sovint, manual. **Galen** ajuda a facilitar aquest procés gràcies a un llenguatge simple, que corre a sobre de Selenium, que permet especificar les expectatives sobre l'aparença de la pàgina web a pantalles de diverses mides. Tot i que Galen té els mateixos problemes de fragilitat i de velocitat típics de qualsevol altre mètode de proves continues, trobem que els primers comentaris sobre disseny són molt beneficiosos.

El fet de poder gestionar els secrets de manera segura s'està convertint en un problema cada cop més gran pels projectes. L'antiga pràctica de guardar els secrets en un arxiu o en variables d'entorn s'està fent cada cop més difícil de gestionar, especialment en entorns amb múltiples aplicacions i un gran nombre de microserveis. **HashiCorp Vault** soluciona aquest problema proporcionant mecanismes per accedir als secrets a través d'una interfícies unificada. Ens ha sigut de gran ajuda en un bon nombre de projectes i als nostres equips els ha agradat com n'era de fàcil d'integrar Vault als seus serveis. Emmagatzemar i actualitzar secrets és una mica incòmode de fer ja que depèn d'una eina de línia d'ordres i una gran quantitat de disciplina per part de l'equip.

Hi ha cada vegada més projectes que emeten i consumeixen informació formatada en JSON. Escriure proves en Java per a JSON pot ser laboriós. **JSONassert** és una petita llibreria creada per ajudar a escriure proves més petites que s'encarreguin de JSON simplificant assercions i proporcionant millors missatges d'error.

Pa11y és un provador d'accessibilitat automàtic que pot córrer des de la línia de comandaments i es pot incrustar en un canal. Els nostres equips han tingut èxit utilitzant Pa11y en una pàgina molt dinàmica creant, en primer lloc, una versió estàtica en HTML i, després, passant les proves d'accessibilitat. Per a molts sistemes, especialment pàgines governamentals, és necessari córrer proves d'accessibilitat i Pa11y ho fa molt més fàcil.

Tenint eines tan bones com Vault, no hi ha excusa que valgui per guardar secrets en els repositoris de codi, especialment quan aquests acostumen a ser el punt feble de sistemes importants. Ja hem mencionat anteriorment

poca amplada de banda o poca bateria.

Els nostres equips han tingut èxit amb **axios**, un client HTTP basat en peticions a JavaScript, que descriuen com "millor que Fetch." El projecte té molts seguidors i és molt actiu a GitHub i, a més, té el nostre vist i plau.

Amb l'augment en l'interès per arquitectures de transmissió de dades en temps real i els oceans de dades que alimenten hem vist un increment en la dependència d'eines de "captura de canvi de dades" per a connectar dipòsits transaccionals de dades a sistemes de processament de transmissions. **Bottled Water** és una incorporació molt benvinguda a aquest camp ja que converteix els canvis al registre d'escriptura anticipada de PostgreSQL en esdeveniments Kafka

Si s'ha de crear una aplicació de pàgina única (SPA per les seves sigles en anglès) i no se sap quin entorn de treball utilitzar, **Ember.js** s'està convertint en la opció més destacada. Als nostres equips els ha agradat molt la gran productivitat de desenvolupament que permet i la menor quantitat de sorpreses en comparació amb altres entorns de treball com AngularJS. La interfície de línia d'ordres de Ember és un paradís entre les eines de desenvolupament JavaScript. A més, l'equip central de Ember i la seva comunitat són molt actius.

Ara que les aplicacions d'una sola pàgina en JavaScript són cada vegada més complexes, estem veient que és cada vegada més important que la gestió de l'estat del client sigui predictible. **Redux**, amb els seus tres principis de restriccions per actualitzar l'estat, ha demostrat ser d'un valor incalculable en alguns dels projectes que hem implementat. Els tutorials Getting Started with Redux i idiomatic Redux són un bon la

l14 TD[més)0.5 t les

immutables. Més equips estan utilitzant aquesta llibreria per ubicar mutacions i per mantenir l'estat en la producció. Recomanem als desenvolupadors que investiguin aquesta llibreria, especialment quan es

ThoughtWorks és una consultoria i comunitat tecnològica formada per persones apassionades i mogudes per objectius. Ajudem els nostres clients a posar la tecnologia al centre del seu negoci i, junts, creem els programes que més els importen. Ens dediquem al canvi social positiu: la nostra missió és crear una humanitat millor a través de programaris i ens associem amb moltes organitzacions que comparteixen els mateixos objectius.

Fundada fa més de 20 anys, ThoughtWorks ha crescut fins a convertir-se en una empresa formada per més de 4000 persones i amb una divisió de productes encarregada